

Appendix VIII — Hibernate¹

An increasing portion of personal computers (PCs) sold today include a “hibernate” feature which saves the system memory state to non-volatile memory (usually the hard disk) and then turns off the system. When the system comes out of this state, the memory image must be read off the disk and basic device reconfiguration done. This is considerably more complex and time-consuming to accomplish than entering or leaving typical sleep states. However, as operating systems become more reliable, it becomes an increasingly attractive state to use to maximize energy savings and battery life. Unfortunately, the concept is ill-understood by most people, and likely to be confusing. User manuals and operating systems present hibernate in a variety of inconsistent ways. Rather than wait till the problem emerges as a large one for the industry, it makes sense to solve it now and we undertook to try to do just that. The goal is to arrive at a common, simple, and consistent presentation of the hibernate state to ordinary PC users. One solution stands out as the simplest and cleanest — that hibernate is a form of *off*.

1.0 Introduction

One of the parts of the Power Control User Interface Standard that elicited the most concern among manufacturers is the specification that “*hibernate*” be clearly identified as a form of *off*. Among those people who were presented with the question, almost all fell into three categories:

- Hardware professionals — people who work with the electrical details of PC hardware. They mostly saw *hibernate* as a form of *sleep*.
- Usability professionals — people who deal with making PCs easier to use. They uniformly agreed that *hibernate* is *off*.
- Everyone else. When the issue was explained, they generally agreed that *hibernate* is *off*, though few of these people would have considered the issue previously, and would not be likely to have a firm opinion.

Faced with this lack of consensus (particularly for the hardware people whose views on safety issues are taken most seriously), we prepared a discussion of the advantages and disadvantages of each design solution. The intent was to fairly represent all views, with the hope to gain a consensus around one solution.

The process began with a first draft from LBNL that was then circulated to the PAC and key other individuals. The last table of the draft included a ranking system to rate the problems that each solution exhibits to show the degree to which each are problematic. The intent was to obtain close review of the discussion and incorporate comments into the text and the rating table. Several people provided verbal comment, which has been incorporated, but no one provided the ratings. Most of the “hardware” people didn’t change their minds (some did), though how much attention they gave to the discussion is not known.

For clarity, operating modes (as the user perceives them) are italicized, e.g. the *off* mode versus an LED being off.



¹ This appendix provides detailed background information about the development of the Power Control User Interface Standard. For the full report and more about the Standard, see <http://eetd.LBL.gov/Controls>

2.0 Dissecting The Issue

This appendix presents our analysis of the hibernate problem. It includes the goals of classifying internal system states to externally perceived ones, six possible solutions to the hibernate problem, and how each solution fares with respect to eleven potential problems. None of the solutions is perfect, but they vary considerably in the number and severity of their problems.

2.1 Context

This discussion presumes as accepted (except where otherwise noted) the other five core pieces of the user interface standard:

- Use three power states (*on*, *off*, and *sleep*);
- Use the term “Power” (for buttons and indicators);
- Use **Green** / **Amber** / Off for power indicators;
- Change the international “standby” symbol —  — to mean “Power”; and
- Use the sleep metaphor and moon icon —  .

At present, the issue is only of major concern for personal computers (PCs), because they are the only devices that have a complex system state *and* commonly restart the operating system. Many devices remember some context between *on* states (e.g. a TV remembering the channel being viewed), but the state information is simple and easily saved in non-volatile memory. Devices such as PDAs are only rebooted when a serious error occurs, not in conjunction with normal on/off cycles. Thus, PDAs lack a normal *off* state other than hibernate. This discussion is organized around PCs (desktop and mobile) running on ACPI and the Windows operating system (version XP or earlier) but the principles should apply to any computer operating system, and ultimately any device.

This “hibernate problem” reduces to assigning ACPI states to user-perceived power states. Possible machine states (in this case for PCs or any device) are shown in Table 1.

Table 1. Possible device states

State	ACPI State(s)	Comments
Active / Full-on	S0	Processing
<i>On</i>	S0	Waiting for input
Resting	S1 or S2	Screen dim
Light Sleep		Faster recovery than <i>Sleep</i>
<i>Sleep</i>	S3	
Deep Sleep		Slower recovery than <i>Sleep</i>
<i>Hibernate</i>	S4 or Mech. Off	
<i>(Soft) Off</i>	S5	
<i>(Hard) Off</i>	Mech. Off	Unplugged, any battery dead or removed

As one moves down the scale, capability, responsiveness, and power consumption all drop. It is unlikely that any machine would have all of these states. It is possible that the assignment of internal states to user-perceived states will eventually vary from system to system, but the goal would be to hide this fact from the user.

Criteria that should be considered in allocating internal system power states to user-perceived states include:

- Indicator light status
- Behavior:
 - Wake events: (responsiveness to buttons, switches, keyboard/mouse input, network activity, etc.)
 - Noise made by the machine (e.g. fans, disks)
 - Recovery time to a full-on state
- Power consumption (*W*)
- Ability to unplug without bad consequences
- Ability to modify internal hardware (e.g. PCI cards, memory, disks)
- Ability to modify external hardware (e.g. USB devices, PC cards, docking station)

The ACPI specification addresses this issue in Table 2-1 (ACPI 2.0 specification, 2000), and is reproduced here as Table 2. The ACPI specification is ambiguous about hibernate, sometimes calling it a sleep state, other times making clear that it is off, and at other times suggesting that it occupies a system state in addition to those shown here². By this table, hibernate differs from sleep in latency and power consumption. Tellingly, hibernate can occur in G2/S5 or G3, that is, with or without the system energized with power while off. It differs from each of these states *only* by the “OS restart required” criterion.

Table 2. Summary of Global Power States (from ACPI 2.0 Specification)

Global system state	Software runs	Latency	Power consumption	OS restart required	Safe to disassemble computer	Exit state electronically
G0 Working	Yes	0	Large	No	No	Yes
G1 Sleeping	No	>0, varies with sleep state	Smaller	No	No	Yes
G2/S5 Soft Off	No	Long	Very near 0	Yes	No	Yes
G3 Mechanical Off	No	Long	RTC battery	Yes	Yes	No

In addition, it should be kept in mind that at present machines are usually turned *on* with a

² The ACPI specification itself uses neither the term “hibernate” nor “standby”.

power button, and *off* with operating system interaction. However, with greater use of *hibernate*, the power button may be increasingly used to go to the *hibernate* form of *off*.

If a user has moments before put a system into hibernate or off, most of the time they will remember which was used (though not always, particularly if it is not their usual computer). The cases that are the most likely to raise issues are when the machine is encountered much later (perhaps days or weeks), or by a different person.

The goal is to identify a set of principles that result in machines that are as simple as possible for people to understand and use while not compromising capabilities.

2.2 Possible solutions

The six solutions shown in Table 3 span the range of reasonable solutions to the “hibernate problem”. We believe that no other such solutions exist, so that one of these must be chosen (the *status quo* is effectively solution E).

Table 3. Possible solutions to the “hibernate problem”

Solution	Description
A	<i>Hibernate is off.</i>
B1	<i>Hibernate is a fourth mode — the power indicator light indicates hibernate</i>
B2	<i>Hibernate is a fourth mode — the power indicator is off</i>
C	<i>Hibernate is a form of sleep (the power indicator is amber in hibernate)</i>
D	<i>Hibernate and sleep and are both forms of off.</i>
E	<i>Hibernate state assignment and indicator light usage varies by machine, even among those running the same operating system. For example, the power light might be on during hibernate for a desktop PC but off for laptops, or only on for laptops when the lid is open.</i>

As solution E fails the basic criteria of general consistency from device to device, it is not evaluated. Since E potentially includes all of the other solutions, it is a problem for all of the issues below. Solution E resumes that consistency is not possible.

The “hibernate” term should be replaced, though by what hinges on hibernate’s state assignment. For solutions A and D, it should be “off” (versus “shutdown off”). For C it should be “deep sleep”.

2.3 Bad consequences with user mis-understanding of power user interface (UI)

- Failure to resume — From changing internal hardware while in *hibernate* (or *sleep*).
- Energy waste — From not using *sleep* and/or *hibernate* due to user confusion.
- Lost data — From losing system state due to unplugging or battery loss while in *sleep*.
- User confusion — From inconsistent or confusing interfaces. Users may not get the benefit of the power modes and behavior which best matches their needs.

- Manufacturer costs — From customer calls to Technical Support lines and/or bad associations with the product and brand.

2.4 Simple arguments for each solution

- A: Major problems with other solutions; only problem with this one is changing internal hardware in hibernate (already a problem).
- B1: *Hibernate* is sufficiently different from *sleep* to warrant a separate mode.
- B2: *Hibernate* is sufficiently different from *sleep* to warrant a separate mode, but indicator burns energy so turn it off.
- C: State is saved in both *hibernate* and *sleep* so same to user.
- D: We can simplify to just *On* and *Off*

2.5 Issues

The following issues are ones that might be of concern in deciding what to do about hibernate.

Simplicity and Consistency of Power UI

The “at most three states” principle is violated

B1 and B2 both require that user’s understand that there is a fourth basic system state. Adding a fourth state adds complexity to people’s mental models and indicator implementation.

The principle that there is a 1:1 correspondence between states and the power indicator is violated

In B2, both *hibernate* and *off* are both indicated by off.

The default “Off” state (from power button) will vary across machines

This may be correlated to whether it is a desktop or laptop, and is already user-selectable. This is true for any of the solutions.

The principle that responsiveness to input is consistent within a state is violated

In C and D, a PC will have different wake events and different recovery times depending on whether it was internally in *sleep* or *hibernate* (or *off*). (A solution to part of this is to remove sleep buttons and disable any wake event from *sleep* other than pressing the power button, but the recovery time difference remains).

The power indicator will be more complicated

B1 requires an additional indication method to show a fourth basic state.

What can the user do without turning a machine on

In one sense these issues are not problems in that the *power state* (*on*, *off*, or *sleep*) and *system state* (booted up or shut down) are different concepts. It is not the function of the power indicator to show the *system state*; the power indicator shows the *power state*, not how the machine got there. Users are accustomed to correlating the two, but if that can be broken (as with a PDA), this is not a problem.

One can't tell from the power indicator if the system's state is saved

This is a problem with A, B2, and D. The system must be woken/resumed/turned-on to determine what the state was. This is only a problem if users rely on the power indicator *alone* to decide if the machine can be opened up and internal components changed.

One can't tell from the power indicator if the machine can be unplugged

This is a problem with C and D — in both cases *sleep* and *hibernate* look the same, and with D, *off* also looks the same.

One can't tell from the power indicator if it is OK to change internal hardware

This is a problem with A, B2, and D. It is also a problem with B1 and C if the battery runs out, is replaced, etc.

For B1 and C, if there is a power outage or the machine is unplugged, then plugged back in, the indicator should come on. This might require extra hardware to implement.

Safety instructions specify that the system should be shut down, unplugged, and any battery removed. So, this is not a safety problem, so long as people follow instructions.

One can't tell from the power indicator if it is OK to change external hardware

Examples are USB devices, PC cards, and notebook docking stations. Whether this is an issue is likely to vary across devices and over time.

Other

The hibernate indicator will run down the battery

This is a problem for B1 and C. It could be mitigated by an intermittent flash, but would still be a problem.

The power consumption can't be inferred from the indicator light

This is a problem with C and D if sleep power is much different than hibernate power. It is assumed that hibernate power = off power.

Machine behavior differs when the power control is a rocker (to zero power) switch, not a button

B1, C, and D can't be implemented.

Machine behavior differs when there is a rocker switch (to zero power) in addition to a power button

B1 and C won't indicate *hibernate* when main power is off. With C and D, you can't tell if it is OK to turn the rocker switch to off. Because of these problems, *hibernate* is not likely to be implemented with these solutions and both a rocker switch and power button on the device.

3.0 Emerging Issues

The context of evaluating *hibernate* is always evolving. The recovery times from *sleep*, *shutdown*, and *hibernate* are all changing with memory sizes, disk and processor speeds, operating systems, configurations, and the availability of non-volatile main memory. In addition, some of the issues discussed above could be mitigated if a *mechanical* indicator of the *hibernate* state was included so that power would not be required to maintain it.

4.0 Results and Conclusions

Table 4 summarizes the issues reviewed in Section 2.5, provides severity levels for each issue, and the total degree of problem presented by each of the five solutions, weighted by the severity, and a simple count. Four of the solutions are fairly close in the degree of problem they present, particularly when seen from the weighted perspective. Only one solution stands out as least problematic — Solution A.

Hibernate should be seen as a form of *off*. This solution (A) has the fewest problems for users, manufacturers, and energy consumption.

5.0 References

Compaq, Intel, Microsoft, Phoenix Tech., and Toshiba. 2000. Advanced Configuration and Power Interface Specification: Revision 2.0. [<http://www.acpi.info>]. 2000.

Table 4. Summary of Solutions/Issues and Severity Ratings

Issue	A	B1	B2	C	D	Severity
<u>Simplicity and Consistence of Power UI</u>						
The “at most three states” principle is violated		X	X			3
No 1:1 correspondence between states and power indicator			X			3
The default “Off” state (from power button) will vary	X	X	X	X	X	1
Responsiveness to input is not consistent within a state				X	X	3
The Power indicator will be more complicated		X				2
<u>What can the user do without turning a machine on</u>						
Can’t tell from the power indicator if the system’s state is saved	X		X		X	2
Can’t tell from the power indicator if machine can be unplugged				X	X	2
Can’t tell from the power indicator if OK to change internal hardware	X	*	X	*	X	1
Can’t tell from the power indicator if OK to change external hardware	?	?	?	?	?	0
<u>Other</u>						
The hibernate indicator will run down the battery		X		X		3
The power consumption can’t be inferred from the indicator light				X	X	1
Behavior differences when the power control is a rocker (to zero power)		X		X	X	1
Behavior differences when there is a rocker switch <i>in addition</i> to a button		X		X	X	1
Total (as if problems were of equal severity)	3	6+	5	7+	8	
Totals by severity rankings	4	11+	10	12+	12	

Severity: 1 = minor concern; 3 = major concern; * = possibly a problem after power failure; ? = not sure if a problem